AP-410

APPLICATION NOTE

Enhanced Serial Port on the 83C51FA

BETSY JONES
ECO APPLICATIONS ENGINEER

November 1987

Order Number: 270490-001

Contact your local Intel sales office or your distributor to obtain the latest specifications before placing your product order.

Copies of documents which have an ordering number and are referenced in this document, or other Intel literature, may be obtained from:

Intel Corporation
P.O. Box 7641
Mt. Prospect, IL 60056-7641

or call 1-800-879-4683

The serial port on the 8051 has been enhanced on the 83C51FA with the addition of two new features: Automatic Address Recognition and Framing Error Detection. **Automatic Address Recognition** facilitates multiprocessor communications by reducing CPU overhead. **Framing Error Detection** increases communication reliability by checking each reception for a valid stop bit.

This Application Note explains how to use these new features with samples of code for typical applications. A section is also included which reviews how to set up the serial port for multiprocessor applications.

MULTIPROCESSOR COMMUNICATIONS

In applications where multiple controllers jointly perform a task, the master controller must be able to communicate selectively with individual slaves. To do this, the master first identifies the target slave (or slaves) with an address byte and then transmits a block of data. The target slaves must be able to identify their own address before receiving any data bytes.

The serial port on the 8051 provides a 9-bit mode to facilitate multiprocessor communication. The 9th bit allows the controller to distinguish between address and data bytes. In this mode, a total of 11 bits are received or transmitted: a start bit (0), 8 data bits (LSB first), a programmable 9th bit, and a stop bit (1). See Figure below.

The 9th bit is set to 1 to identify address bytes and set to 0 for data bytes. A typical data stream is seen below:

ADDRESS BYTE / DATA BYTE / DATA BYTE / . . .
D8 = 1 D8 = 0 D8 = 0

Initially the slave is set up to only receive address bytes. Once it receives its own address, the slave reconfigures itself to receive data. On the 8051 serial port, an address byte interrupts **all** slaves for an address comparison. On the 83C51FA, however, Automatic Address Recognition allows the addressed slave to be the **only** one interrupted; that is, the address comparison occurs in hardware, not software. With this feature, the master controller can establish communication with one or more slaves without all the slaves having to respond to the transmission.

AUTOMATIC ADDRESS RECOGNITION

Automatic Address Recognition reduces the CPU time required to service the serial port. Since the CPU is only interrupted when it receives its own address, the software overhead to compare addresses is eliminated. This would also effectively reduce the sophistication of the serial protocol when numerous controllers are sharing the same serial link.

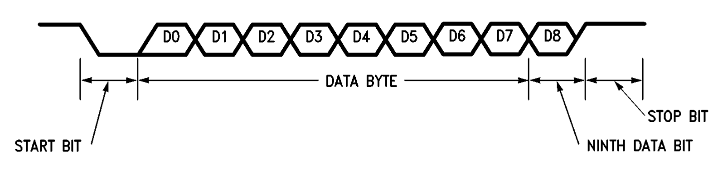
This same feature can also be used in conjunction with the Idle Mode to reduce the system's overall power consumption. For instance, a master may need to communicate with only one slave at a time. With all slaves in Idle Mode, only that one slave would be interrupted to respond to the master's transmission. Without Automatic Addressing, each slave would have to "wake up" to check for its address. Limiting the interruptions reduces the amount of current drawn by the system and thus reduces the power consumption.

In multiprocessor applications the serial port is configured in either of the 9-bit modes (Mode 2 or 3). Mode 2 has a fixed baud rate whereas Mode 3 is variable. For more information on the different serial port modes refer to the "Serial Port Set Up" section.

Automatic Address Recognition is enabled by setting the SM2 bit in SCON. Each slave has its SM2 bit set waiting for an address byte (9th bit = 1). The Receive Interrupt (RI) flag will get set when the received byte corresponds to either a Given or Broadcast Address. The slave then clears its SM2 bit to enable reception of data bytes (9th bit = 0) from the master.

The master can selectively communicate with groups of slaves by using the Given Address. Addressing all slaves at once is possible with the Broadcast Address. These addresses are defined for each slave by two new Special Function Registers: SADDR and SADEN.

A slave's individual address is specified in SADDR. SADEN is a mask byte that defines don't-cares to form the Given Address. These don't-cares allow flexibility in the user-defined protocol to address one or more slaves. The following is an example of how to define Given Addresses and selectively address different slaves.



270490–1

Slave 1

SADDR	=	1111	0001
SADEN	=	1111	1010
GIVEN	=	1111	0X0X

Slave 2

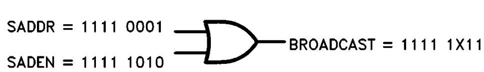
SADDR	=	1111	0011
SADEN	=	1111	1001
GIVEN	=	1111	0XX1

The SADEN bytes have been selected such that bit 1 (LSB) is a don't-care for Slave 1's Given Address, but bit 1 = 1 for Slave 2. Thus, to selectively communicate with just Slave 1 an address with bit 1 = 0 would be used (e.g. 1111 0000).

Similarly, bit 2 = 0 for Slave 1, but is a don't-care for Slave 2. Now to communicate with just Slave 2 an address with bit 2 = 1 would be used (e.g. 1111 0111).

Finally, to communicate with both slaves at once the address must have bit 1 = 1 and bit 2 = 0. Notice, however, that bit 3 is a "don't-care" for both slaves. This allows two different addresses to select both slaves (1111 0001 or 1111 0101). If a third slave was added that required its bit 3 = 0, then the latter address could be used to communicate with Slave 1 and 2 but not Slave 3.

The master can also communicate with all slaves at once with the Broadcast Address. It is formed from the logical OR of SADDR and SADEN with zeros defined as don't-cares. For example, the Broadcast address for Slave 1 would be formed as follows:



270490–2

The don't-cares also allow flexibility in defining the Broadcast Address, but in most applications a Broadcast Address will be 0FFH.

SADDR and SADEN are located at address A9H and B9H, respectively. On Reset, SADDR and SADEN are initialized to 00H which defines the Given and Broadcast Addresses as XXXX XXXX (all don't-cares). This assures the 83C51FA serial port to be backwards compatible with the other MCS®-51 products which do not implement Automatic Addressing.

FRAMING ERROR DETECTION

Framing Error Detection is another new feature on 83C51FA serial port which allows the receiving controller to check for valid stop bits in Modes 1, 2, or 3. A missing stop bit can be caused, for example, by noise on the serial lines or transmission by two CPUs simultaneously.

If a stop bit is missing a Framing Error bit FE will be set. This bit can then be checked in software after each reception to detect communication errors. Once set, the FE bit must be cleared in software. A valid stop bit will not clear FE.

The FE bit is located in SCON and shares the same bit address as SM0. To determine which is accessed, a new control bit called SMOD0 has been added in the PCON register (see figures below). If SMOD0 = 0, then accesses to SCON.7 are to SM0. If SMOD0 = 1, then accesses to SCON.7 are to FE.

PCON: Power Control Register (Not Bit Addressable)

SMOD1	**SMOD0**	—	POF	GF1	GF0	PD	IDL

Address = 87H

SCON: Serial Port Control Register (Bit Addressable)

SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI

Address = 98H

SERIAL PORT SOFTWARE

The following sections of code show examples of how to invoke Automatic Addressing and Framing Error Detection. Routines for both the slave and master are given. Code is also included to initialize both serial ports; however, for more information on setting up the serial port refer to the next section.

For this example, the master and slave are transmitting/receiving at 9600 baud with a 12 MHz crystal frequency. To obtain this baud rate, the serial port is configured in Mode 3 and Timer 2 is used as the baud-rate generator.

Listing 1 shows the initialization for the slave. Notice that Automatic Addressing and Framing Error Detection are enabled. The Given and Broadcast addresses for this slave are taken from Slave 1 in the previous example. A temporary byte has also been defined to store the incoming data byte.

The slave will remain in Idle Mode until it is interrupted by its own address. At that point, it clears the SM2

Listing 1. Initialization Routine for the Slave

```
ORG   00H
LJMP  INIT

ORG   0023H
LJMP  SERIAL_PORT_INTERRUPT

TEMP  DATA  30H                     ; Temporary storage byte

INIT: MOV SCON, #0F0H               ; Mode 3, enable Auto Addressing
                                    ; and reception
      ORL PCON, #40H                ; FE bit accessed (SMOD0 = 1)
      MOV RCAP2H, #0FFH             ; Reload values for 9600 Baud
      MOV RCAP2L, #0D9H
      MOV T2CON, #34H               ; Timer 2 set up, TR2 = 1 turns
                                    ; timer on

INTERRUPTS:  SETB EA                ; Enable global interrupt
             SETB ES                ; Enable serial port interrupt

ADDRESSES:   MOV SADDR, # 11110001 ; Define Given & Broadcast
             MOV SADEN, # 11111010 ; Addresses
                                    ; GIVEN     = 11110X0X
                                    ; BROADCAST = 11111X11

IDLE_MODE:   ORL PCON, #01H         ; Invoke Idle Mode
```

Listing 2. Receive Routine for the Slave

```
SERIAL_PORT INTERRUPT:
        PUSH PSW
        CLR RI                      ; RI set when address is
                                    ; recognized & must be cleared
                                    ; in software
        CLR SM2                     ; Reconfigure slave to receive
                                    ; data bytes

RECEIVE_DATA:
        JNB RI, $                   ; Wait for RI to be set
        MOV C, SCON.7               ; Check for framing error
        JC FRAMING_ERROR
        MOV TEMP, SBUF              ; Receive data byte & store
                                    ; in temporary location
        CLR RI                      ; Clear flag for next
                                    ; reception
        SETB SM2                    ; Re-enable Automatic
                                    ; Addressing
        POP PSW
        RETI

FRAMING_ERROR:
        CLR SCON.7                  ; Clear FE bit
        CLR C
            •
            •                       ; Error routine left up to
            •                       ; the user
        POP PSW
        RETI
```

Listing 3. Initialization and Transmit Routines for the Master

```
GIVEN_1        equ     11110001B
MESSAGE_1      data    30H

INIT:    MOV SCON, #0D0H             ; Mode 3, REN = 1
         MOV RCAP2H, #0FFH           ; 9600 Baud
         MOV RCAP2L, #0D9H
         MOV T2CON, #34H             ; Timer 2 set up, TR2 = 1

TRANSMIT_ADDRESS:
         CLR TI
         SETB TB8                    ; Mark 1st byte as an address
                                     ; byte (9th bit = 1)
         MOV SBUF, #GIVEN_1          ; Send address
         JNB   TI, $                 ; Wait for transmission
                                     ; complete
         CLR TI                      ; Clear flag for next
                                     ; transmission

TRANSMIT_DATA:
         CLR TB8                     ; Mark 2nd byte as a data
                                     ; byte (9th bit = 0)
         MOV SBUF, MESSAGE_1         ; Send data byte
         JNB TI, $
         CLR TI
```

bit to enable reception of data bytes. Depending on the user's protocol, more than one data byte may actually be received. This example, however, assumes only one byte of data follows each address byte.

Listing 2 shows the receive routine. Notice that when the data byte is received, the software checks for a framing error. The error routine could, for example, send an error message to the master and ask the master to re-transmit the last message. Before exiting the routine the SM2 is set to 1 to reenable Automatic Addressing. Once the slave has responded to the master's command, it could also put itself back into Idle Mode to wait for the next message.

The initialization routine for the master in Listing 3 is very similar to the slave. In this example, however, the master does not need Automatic Addressing; it is simply transmitting address and data bytes. GIVEN_1 is a byte to address the slave in the above example. MESSAGE_1 is a register that contains the data byte sent to this slave. Its value is arbitrary for the sample code.

SERIAL PORT SET UP

This section describes how to initialize the 83C51FA serial port for multiprocessor applications. Two different modes are available which provide 9-bit operation: Mode 2 which has a fixed baud rate and Mode 3 which has a variable baud rate. Baud rates can be generated by either Timer 1 or Timer 2 (available on the 83C51FA but not the 8051). Deciding which mode and timer to use is determined by the desired baud rate and clock frequency of the particular application.

Another consideration is the tolerance needed between serial ports. Since the serial port re-synchs its receiver at every start bit, only 8 or 9 bit-times are available to accumulate timing errors. As a result, the receiver and transmitter only have to be within about 5% of each other's baud rate. Allowing equal error to both transmitter and receiver, only about 2% accuracy is actually needed.

Following is a discussion of both Modes 2 and 3 and examples of how to program each. The mode selection bits (SM0 and SM1) are located in SCON. The REN bit must also be set to enable reception.

SCON: Serial Port Control Register (Bit Addressable)

SM0	**SM1**	SM2	**REN**	TB8	RB8	TI	RI

Address = 98H

Mode	SM0	SM1	Baud Rate
2	1	0	Fosc/64 or Fosc/32
3	1	1	Variable

Example 1. Serial Port Mode 2

```
; Frequency          = 12 MHz
; Desired Baud Rate  = 375 kBaud
;                    = 1/32 (Osc Freq)

MOV SCON, #0B0H    ; Serial port Mode 2
                   ; Automatic Addressing (SM2 = 1),
                   ; reception enabled (REN = 1)
ORL PCON, #80H     ; SMOD1 = 1 to double baud rate
```

Mode 2

Mode 2 uses a fixed baud rate of 1/32 or 1/64 of the oscillator frequency depending on the value of the SMOD1 bit in PCON. This mode basically offers a choice of two high-speed baud rates. With a 12 MHz clock frequency, baud rates of 187.5 kbaud or 375 kbaud can be obtained.

None of the timer/counters need to be set up for Mode 2. Only the SFRs SCON and PCON need to be defined.

PCON: Power Control Register (Not Bit Addressable)

SMOD1	SMOD0	—	POF	GF1	GF0	PD	IDL

Address = 87H

The baud rate in this mode is calculated by:

$$\text{Mode 2 Baud Rate} = \frac{2^{\text{SMOD1}} \times \text{Osc Freq}}{64}$$

SMOD1 = 0, Baud Rate = 1/64 Osc Freq

SMOD1 = 1, Baud Rate = 1/32 Osc Freq

Mode 3

Mode 3 of the serial port has a variable baud rate generated by either Timer 1 or Timer 2. The baud rate is generated by the rollover rate of the selected timer. The timer is operated in an auto-reload mode so it will roll over to the reload value selected in software.

Baud rates based off Timer 2 have less granularity so that almost any baud rate can be obtained at a given clock frequency. However, Timer 1 is sufficient if the desired baud rate can be obtained at the specified clock frequency. Remember baud rates only need about 2% accuracy.

Timer 1 Set Up

To generate baud rates Timer 1 is usually configured in 8-bit auto-reload mode (Mode 2). The mode select bits are M1 and M0 located in TMOD. To turn on Timer 1 the TR1 bit in TCON must be set. Also, the Timer 1 interrupt should be disabled in this application so that when the timer overflows it does not generate an interrupt.

TMOD: Timer/Counter Mode Control Register
(Not bit addressable)

Timer 1				Timer 0			
GATE	C/$\overline{\text{T}}$	**M1**	**M0**	GATE	C/$\overline{\text{T}}$	M1	M0

Address = 89H

TCON: Timer/Counter Control Register
(Bit addressable)

TF1	**TR1**	TF0	TR0	IE1	IT1	IE0	IT0

Address = 88H

The formula for calculating the baud rate is given below. TH1 is the reload value for Timer 1 when it overflows.

$$\text{Baud Rate} = \frac{\text{K} \times \text{Osc Freq}}{32 \times 12 \times [256 - (\text{TH1})]}$$

K = 1 if SMOD1 = 0.

K = 2 if SMOD1 = 1. (SMOD1 is at PCON.7)

If the baud rate is known, the reload value TH1 can be calculated by:

$$\text{TH1} = 256 - \frac{\text{K} \times \text{Osc Freq}}{384 \times \text{Baud Rate}}$$

TH1 must be an integer value. Rounding off TH1 to the nearest integer may not produce the desired baud rate with the 2% accuracy required. In this case, another crystal frequency may have to be chosen.

Refer to Table 1 for timer reload values for commonly used baud rates.

Table 1. Commonly Used Baud Rates Generated by Timer 1

Baud Rate	Osc Freq	SMOD1	Timer 1	
			TMOD	Reload Value
62.5K	12 MHz	1	20	FFH
19.2K	11.06 MHz	1	20	FDH
9.6K	11.06 MHz	0	20	FDH
4.8K	11.06 MHz	0	20	FAH
2.4K	11.06 MHz	0	20	F4H
1.2K	11.06 MHz	0	20	E8H
300	6 MHz	0	20	CCH
110	6 MHz	0	20	72H

Example 2. Serial Port Mode 3, with Timer 1 as Baud-Rate Generator

```
; Frequency          = 11.0 MHz
; Desired Baud Rate  = 19.2 kBaud
;
;                (2) x (11.0 x 10^6)
;  TH1 = 256 - ------------------------
;              (32) x (12) x (19200)
;

;      = 253 = FDH

MOV SCON, #0F0H     ; Serial port Mode 3, SM2 = 1,
                    ; REN = 1
ORL PCON, #80H      ; SMOD1 = 1
MOV TMOD, #20H      ; Timer 1 Mode 2
MOV TH1, #0FDH      ; Reload value for desired baud
                    ; rate
SETB TR1            ; Turn on Timer 1
```

It can be seen that the exact frequency to generate the standard baud rates (19.2K, 9600, 4800, etc.) is 11.06 MHz. However, it is **not** necessary to use this exact frequency. With a 2% tolerance any crystal value from 10.8 MHz to 11.3 MHz is sufficient.

Timer 2 Set Up

Timer 2 has a special baud-rate generator mode which transmits and receives at the same baud rate. This mode is invoked by setting both the RCLK and TCLK bits in T2CON. To turn Timer 2 on the TR2 bit should also be set.

Unlike Timer 1, this mode does not require that the timer overflow interrupt be disabled. That is, when Timer 2 is in the baud-rate generator mode, its interrupt is disconnected from the Timer 2 overflow. This interrupt then becomes available as a third external interrupt. (For more information on external interrupts, refer to the chapter "Hardware Description of the 8051" in the Embedded Controller Handbook.)

T2CON: Timer/Counter 2 Control Register
(Bit Addressable)

TF2	EXF2	**RCLK**	**TCLK**	EXEN2	**TR2**	$C/\overline{T2}$	$CP/\overline{RL2}$

Address = C8H

This formula for calculating the baud rate is given below. (RCAP2H, RCAP2L) is the 16-bit reload value when Timer 2 overflows.

$$\text{Baud Rate} = \frac{\text{Osc Freq}}{32 \times [65536 - (\text{RCAP2H, RCAP2L})]}$$

where (RCAP2H, RCAP2L) is a 16-bit unsigned integer.

To obtain the reload value for RCAP2H and RCAP2L the above equation can be rewritten as:

$$(\text{RCAP2H, RCAP2L}) = 65536 - \frac{\text{Osc Freq}}{32 \times \text{Baud Rate}}$$

Refer to Table 2 for reload values for commonly used baud rates.

Notice that when using Timer 2, most standard baud rates can be obtained at 12 MHz.

Table 2. Commonly Used Baud Rates Generated by Timer 2

Baud Rate	Osc Freq	Timer 2	
		RCAP2H	RCAP2L
375K	12 MHz	FF	FF
9.6K	12 MHz	FF	D9
4.8K	12 MHz	FF	B2
2.4K	12 MHz	FF	64
1.2K	12 MHz	FE	C8
300	12 MHz	FB	1E
110	12 MHz	F2	AF
300	6 MHz	FD	8F
110	6 MHz	F9	57

Example 3. Serial Port Timer with Timer 2 as Baud-Rate Generator

```
; Frequency          = 12 MHz
; Desired Baud Rate  = 9600 Baud
;
;                                  (12 x 10^6)
;    (RCAP2H, RCAP2L) = 65536 -  ---------------
;                                (32) x (9600)
;
;                    = 65497 = FFD9H

MOV SCON, #0F0H        ; Serial port Mode 3, SM2 = 1,
                       ; REN = 1
MOV RCAP2H, #0FFH      ; Reload values for desired
MOV RCAP2L, #0D9H      ; baud rate
MOV T2CON, #34H        ; Timer 2 as baud rate
                       ; generator, turn on Timer 2
```